Financial Machine Learning in Relative Value Arbitrage

Sheikh Sadik

July 2021

1 Data and Period Selection

Our data consists of daily returns collected from CRSP, starting from January 1965 up until December 2020. We restrict our sample with constituents within the SP 500 and trading in NYSE and NASDAQ (exchange code 1 and 3). Our portfolio formation period should consist of stocks with at most five years of data with minimum periods of three years. We select pairs within this training period and let them trade for the next six months. We form portfolios at every six month interval.

Figure 1: An Illustration of Portfolio Formation and Trading Period

2 Candidate Pairs Selection: Unsupervised Approach

We start off with a candidate pairs selection methodology to find a subset of stocks from the entire universe. Traditional methods implement correlation/distance based methods to find a subset of stocks. For example, a commonly used metric is called sum of squared distance. For any given pair of stocks i and j, the sum of squared distance between the two stocks, defined as $SSD_{i,j}$, is calculated as follows,

$$SSD_{i,j} = \frac{1}{T}\sum_{t}(log(P_{i,t}) - log(P_{j,t}))^2 \tag{1}$$

$P_{i,t}$ is the price of the i^{th} stock at time t. Given a universe of N assets, the idea is to calculate the aforementioned metric for each of the $\frac{N(N-1)}{2}$ pairs and rank the pairs in ascending order. The pairs ranked the lowest are selected as ideal pairs for further analysis. However, there are a few caveats to this approach.

First, this is a method purely based on price correlation, which can lead to spurious relationship due to the non–stationary nature of prices. Furthermore, solely using this metric as a pair selection method can lead to pairs with the potential absence of long term equilibrium, which can lead to massive price divergence. Price distance based metrics are also sub-optimal in nature. Given that an ideal pair selected based on distance

method will constitute zero squared distance, which in turn means no spread between the two assets, which in turn may be a profitable trading strategy due to low variance.

Second, we face the issue of working with a high dimensional universe. A portfolio manager needs to search an entire universe of N assets and calculate this metric for potential $\frac{N(N-1)}{2}$ pairs. Imagine having to search from an universe of 20,000 stocks, this equates to 199,990,000 potential pairs!

Our machine learning based approach mitigates these caveats through a two stage clustering process. First, we apply agglomerative clustering on the distance matrix computed from the correlation matrix (from daily returns) computed during the portfolio formation period. This way each stock gets assigned to an individual cluster. The clustering process enables us to classify stocks which are closer to each other, as defined by their pairwise distance. The clusters enable us to create subsets of our universe based on the common variations or factor structures amongst the stocks. Now, within each cluster, we apply graphical lasso on the denoised covariance matrix. The graphical lasso identifies pairs which are conditionally correlated, thus providing us with a sparse representation of the covariance matrix within each cluster. This process minimizes the number of pairs we need to investigate for possible co-integrated relationships.

2.1 Agglomerative Hierarchial Clustering (AHC)

At each training period, the first step in our candidate pairs selection process starts off with applying agglomerative clustering. For a given $N \times N$ correlation matrix $\boldsymbol{R}$, with N being the total number of stocks, we compute the distance matrix $\boldsymbol{D}$ as follows,

$$\boldsymbol{D} = \sqrt{\frac{1}{2} * (1 - \boldsymbol{R})} \tag{2}$$

Each element of $\boldsymbol{D}$ is defined as $d_{i,j}$, which is the distance between stock i and stock j, with $d_{i,j} \in [0, 1]$.

Next, we apply the AHC algorithm on the distance matrix. The algorithm for AHC starts with each stock being its own cluster, then sequentially merges with other stocks which demonstrates a higher degree of similarity or a lower degree of dissimilarity. The dissimilarity measure used here is euclidean distance. The euclidean distance between two column vectors i and j of $\boldsymbol{D}$, define as $\hat{ecd}_{i,j}$ can be calculated as,

$$ecd_{i,j} = \sqrt{\sum_{n} (d_{n,i} - d_{n,j})^2} \tag{3}$$

Furthermore, we use single-linkage method to merge any two sets into one cluster. The single-linkage method merges two sets based on the minimum dissimilarity $ecd_{i,j}$ over all possible cluster pairs. The co-ordinates for the merged sets, denoted as $\hat{i}$ and $\hat{j}$ is computed as,

$$(\hat{i}, \hat{j}) = \arg\min_{i,j} \{ecd_{i,j}\}_{i \neq j} \tag{4}$$

Instead of relying on an optimal cluster selection methodology like the elbow method or gap statistic, we simply set the number of clusters to 5 for computational efficiency. This way, the algorithm cuts off the dendogram at the height point at where we retain 5 unique clusters. AHC enables us to work with smaller subsets of the universe where each asset within a cluster can be thought as sharing a common factor structure based on their proximity as defined by their correlations. However, for two assets to have a relative value proposition, one may want to focus on the conditional correlation, as opposed to their unconditional pairwise

correlation. This can be due the unconditional correlation primarily dominated by the common factor structure of the cluster/universe, as opposed to the pairs sharing common idiosyncratic variation. Thus, for next step, within each cluster, we apply the graphical lasso and retain a sparse representation of the dependence structure of stocks contained within each cluster.

2.2 Graphical Lasso (GL)

Within each cluster, we apply graphical lasso on the empirical covariance matrix and estimate a sparse inverse covariance/precision matrix. The partial correlation matrix derived from this precision matrix can be interpreted as a measure of conditional correlation of two stocks within each cluster.

For a given cluster c, the $N_c \times N_c$ covariance matrix $\boldsymbol{\Sigma_c}$ has an inverse covariance matrix denoted as $\boldsymbol{\Theta_c} = \boldsymbol{\Sigma_c^{-1}}$. The GL algorithm estimates a sparse precision matrix denoted as $\hat{\boldsymbol{\Theta}}_{\boldsymbol{c}}$ by maximizing the following log likelihood function,

$$\hat{\boldsymbol{\Theta}}_{\boldsymbol{c}} = \arg\max \left[\log det(\boldsymbol{\Theta_c}) - tr(\boldsymbol{\Sigma_c}\boldsymbol{\Theta_c}) - \lambda||\boldsymbol{\Theta_c}||_1\right] \tag{5}$$

Here, $\log det(\boldsymbol{\Theta_c})$ is log determinant of the inverse covariance matrix, $tr(\boldsymbol{X})$ is the trace of an input matrix $\boldsymbol{X}$, $||\boldsymbol{\Theta_c}||_1$ is the L1 norm of the inverse covariance matrix, and λ is the regularization parameter.

Next, we compute the partial correlation matrix $\boldsymbol{P}$ from the estimated sparse precision matrix. Let us define $\boldsymbol{\phi}$ to $N_c \times 1$ vector, with $\boldsymbol{\phi} = diag(\hat{\boldsymbol{\Theta}}_{\boldsymbol{c}})^{-0.5}$. $\boldsymbol{P}$ can be computed as,

$$\boldsymbol{P} = -\boldsymbol{\phi}\boldsymbol{\phi'} \circ \hat{\boldsymbol{\Theta}}_{\boldsymbol{c}} \tag{6}$$

For a given training period and for each cluster, we estimate a sparse precision matrix through a cross-validation procedure. The cross validation procedure also obtains the optimal regularization parameter λ. The cross validation procedure is a variation of k-fold cross validation but catered towards time series data. We explain the cross validation procedure in details at a later section. We set the number of folds to 4.

Now, armed with a partial correlation matrix $\boldsymbol{P}$, we rank all the pairs in descending order. Each pair with 0 partial correlation represents two stocks that are conditionally independent. The idea behind this can be explained in a network graph context. If we visualize a correlation matrix in an undirected network graph, each vertex of the graph will represent a stock and each stock is connected to all the other stocks through their corresponding edges. When we apply graphical lasso to a covariance matrix, it greatly reduces the number of edges one vertex can be connected to. The absence of an edge between two vertices will imply that there is no conditional dependence between them and vice versa. For our analysis, We discard all the pairs with partial correlation less than 0.10. This way we only retain pairs which are conditionally linked to each other and discard most of the conditionally independent pairs or pairs with weak conditional dependence. We use this final list of pairs obtained from each cluster for the next stage of pairs selection process.

3 Final Pairs Selection: Cointegration and Mean Reversion

3.1 Cointegration Tests

Our proposed unsupervised learning method as outlined above is based on the underlying covariance structure of our data. However covariance related measures usually capture short run dynamics in a system and won't inform us about the long run dynamics of assets. As for this reason, we move on testing whether the selected

pairs from the previous section demonstrate long run relationships. If the spreads calculated between two assets demonstrate a stationary process, then they are related to each other over the long run through a common stochastic trend, and so we can say that they are cointegrated. In the traditional framework of relative value strategies, particularly in equities, testing for possible cointegration between two assets before designing a trade is crucial, since if there is no evidence of long run equilibrium, the spreads can widen indefinitely, resulting in a nonprofitable strategy.

The most widely used test for standard cointegration is the Engle-Granger 2-step test. Under this case, we are looking for where $\{log(P_i)\}_t$ and $\{log(P_j)\}_t$ are both $I(1)$, there may exist a linear combination between the two log prices such that their linear combination is $I(0)$. One caveat here is that asset prices have memory and so does the spread between two asset prices. In contrast, standard cointegration requirements between a pair of assets may lead to complete cut off of memory. This may not be feasible especially in the case of investment strategies which take advantage of equilibrium models, since some memory is required to assess how far the spread between two assets have drifted away from their long term equilibrium relationship. Thus, there is a minimum amount of differentiation, denoted as d, where d is non-negative real number and $d < 1$, which will make a price series stationary while preserving some of the memory. This transformation is known as fractional differencing. In the case of equilibrium models, two fractionally differentiated assets can demonstrate a cointegrated relationship where their spreads are shown to have long memory and stationary at the same time. Thus, two price processes are fractionally cointegrated if both processes are $I(d)$ and there exists a linear combination with $I(d_u)$, with $d_u < d$.

In the next subsection, we briefly outline the Engle-Granger 2-step test. Next, we propose another cointegration test to our analysis based on fractionally cointegrated residuals.

3.1.1 Engle Granger 2-Step Cointegration Test

If the log prices of a pair of stocks, defined as $\{log(P_{i/j})\}_t$ are unit root processes as in they are $I(1)$, we want to test if they are cointegrated. The two step approach is outlined as follows, 1) For any given pair i and j, we first run the following regression,

$$log(P_{i,t}) = \alpha + \beta log(P_{j,t}) + \epsilon_t \tag{7}$$

We extract the residuals ϵ_t and conduct an augmented dickey fuller test using the following specification,

$$\epsilon_t = \gamma \epsilon_{t-1} + \sum_{i=1}^{K} c_i \Delta \epsilon_{t-i} + \eta_t \tag{8}$$

The null hypothesis, $H_0 : \gamma = 0$, is of no cointegration against the alternative hypothesis $H_a : \gamma < 0$.

2) Next step, we fit an error correction model with the error correction term defined as the residuals extracted from (7)

$$\Delta log(P_{i,t}) = \delta + \phi \epsilon_{t-1} + \lambda \Delta log(P_{i,t-1}) + \kappa_0 \Delta log(P_{j,t}) + \kappa_1 \Delta log(P_{j,t-1}) + u_t \tag{9}$$

We can also estimate a similar error correction model for P_j. Error correction requires that $\phi < 0$, which means if the error correction term was positive in the last period, which means $log(P_i)$ was above the equilibrium value of $log(P_j)$ and thus a negative ϕ will bring $log(P_i)$ back to the equilibrium value. We can also conduct standard t-tests on the coefficients. In all steps, we use a 5% significance level.

3.1.2 Hurvich-Chen Fractional Cointegration Test

As for our fractional cointegration test, we employ the semiparametric residuals based test proposed by Chen, Hurvich (2003). Let, $\boldsymbol{X}$ be a $T \times N$ matrix ($N = 2$ in our case) with $\boldsymbol{X} = (\boldsymbol{p_i}, \boldsymbol{p_j})$ with $\boldsymbol{p_i}$ being $T \times 1$ vector of log prices $log(P_{i/j})$ corresponding to the i^{th} stock. Now, at time t, for a given vector of log prices for the pairs, denoted as $\xi_t = [log(P_{i,t})\, log(P_{j,t})]'$,

1) Let us calculate the tapered discrete Fourier transformation as,

$$J_\xi(\omega_f) = \frac{1}{\sqrt{2\pi \sum |h_t^{p-1}|^2}} \sum_t^{T/2} h_t^{p-1} \xi_t e^{iw_f t} \tag{10}$$

Where, $\omega_f = \frac{2\pi f}{T/2}$ is the fth fourier frequency and h_t is the complex valued taper,

$$h_t = \frac{1}{2}(1 - e^{i2\pi t/(T/2)}), t = 1, ..., T/2 \tag{11}$$

Here, $p = 1$ yields the no-tapering case.

2) Next, we compute the $(N \times N \times T/2)$ tapered cross periodogram matrix of two vector sequence $\{\xi_t\}_t^{T/2}$ and $\{\zeta_t\}_t^{T/2}$ by,

$$I_{\xi\zeta} = J_\xi(\omega_f) J_\zeta^*(\omega_f)' \tag{12}$$

$J_\zeta^*(\omega_f)$ is the complex conjugate matrix of $J_\xi(\omega_f)$.

3) Next, we compute the $(N \times N)$ averaged real periodgram matrix of $T/2$ observations $\{\xi_t\}_t^n$,

$$I_m = \sum_{j=1}^{m} Re\{I_{\xi\zeta,j}\} \tag{13}$$

Here, we set m to 25.

4) Now, we conduct an eigenvalue decomposition of I_m and sort the eigenvectors in ascending order of their eigenvalues. Let, $\boldsymbol{\Gamma}$ be $N \times N$ matrix of sorted eigenvectors based on the corresponding eigenvalues. The $T \times N$ cointegrated residual matrix can be calculated as,

$$\boldsymbol{W} = \boldsymbol{X\Gamma} \tag{14}$$

5) Next, we estimate the memory parameters, denoted as d_i, for each vector of cointegrated residuals in $\boldsymbol{W}$. The memory parameters are estimated using the Gaussian semiparametric estimator [Robinson (1995b)].

6) After estimating the memory parameters, let us define a $N \times 1$ vector of memory parameters $\boldsymbol{d}$ where each element corresponds to each vector of cointegrated residuals from matrix $\boldsymbol{W}$, with $\boldsymbol{d} = [d_1 ... d_N]$. Under the null hypothesis of no cointegration, we define the test statistic as $T = \sqrt{m_n} * (d_N - d_1)$. The null hypothesis will be rejected in favor of fractional cointegration if and only if $T > (\Phi_p/2)^{1/2} z_{\alpha/2}$, with $\Phi_p = \frac{\Gamma(4p-3)\Gamma^4(p)}{\Gamma^4(2p-1)}$. We set m_n to $T^{0.65}$ and α to 5%.

For each of the candidate pairs obtained through the unsupervised method, we apply both cointegration tests as described above. We discard all the pairs which fail to pass at least one of the cointegration tests. The selected cointegrated pairs are passed to the next filtering stage, where we assess the cointegrated spreads' persistence and mean reversion speed.

3.2 Mean Reversion Assessment

In this section, we take the pairs selected from the proposed cointegration tests and conduct additional validation test to assess the mean reverting characteristics of the spreads. cointegration tests only gives us an answer with regards to the stationary nature of the spreads and whether there exists long run dynamics in a pair. However, this doesn't inform us on the degree at which we will expect the spread to converge to its long term equilibrium and how long will it take for the convergence to manifest. For this reason, we assess the pairs' mean reversion characteristics based on two different metrics, the Hurst exponent and the half life of mean reversion.

Hurst Exponent: The Hurst exponent is a measure of long term memory of financial time series. This method measures how much a time series' speed of diffusion is different from the geometric brownian motion assumption. The calculation of Hurst exponent is influenced by rescaled range analysis, which assesses how much the variation of time series changes with length of period being considered. The Hurst exponent is directly related to fractal dimension, which in a nutshell measure the intensity of randomness of a price process.In financial market application, this metric was popularized by Benoit Mandlebrot through his work on chaos theory and fractal analysis [need citations].

Let's denote ϵ_t as the spread obtained from two cointegrated log price series. Let us define $X_t = \Delta\epsilon_t$. We take the time series of spread with length T and divide them into shorter length of k with $k = T, T/2, T/4, ...$ For a given length of k, we iterate through the following steps,

1) We calculate the mean of the spread change.

$$\mu_X = \frac{1}{k}\sum_{i}^{k} X_i \tag{15}$$

2) Next, we demean X_i and denote it as $X_{dm,i}$. Using these demeaned series, we compute the standard deviation and the cumulative series, denoted as $\sigma(k)$ and Z_t,

$$Z_t = \sum_{i}^{k} X_{dm,i} \tag{16}$$

3) We calculate the range of this cumulative series, denoted as $R(k)$,

$$R(k) = max(Z_1, Z_2, ..., Z_k) - min(Z_1, Z_2, ..., Z_k) \tag{17}$$

4) Next, the rescaled range for a given subset of series with length k, is calculated as $\frac{R(k)}{\sigma_k}$, then the expected rescaled range $\mathbb{E}\,\frac{R(k)}{\sigma_k}$ for a given k is simply the average of all possible time series subsets of length k.

5) In order to estimate the Hurst exponent, we run the following regression,

$$\log\left[\mathbb{E}\,\frac{R(k)}{\sigma_k}\right] = c + H\log[k] + \eta_k \tag{18}$$

Here, the coefficient H is the Hurst exponent. If $H < 0.5$, the time series in question tends be more mean reverting. With an $H > 0.5$, the metric suggesting trendiness in the underlying process. Finally, with $H = 0.5$, the underlying process is simply follows geometric brownian motion. For our analysis, we are interested in cointegrated pairs with $H < 0.5$.

Half-Life of Mean Reversion: The original concept of half-life comes from physics, to find the rate of decay of a particular substance. The half-life HL of the spread between two log prices, denoted as ϵ_t, is the time it will take for the expected value of ϵ_t to reach a level between its current value and its long run mean μ_ϵ. Let's assume the price spread follows the Ornstein-Uhlenbeck process:

$$d\epsilon_t = \theta(\mu_\epsilon - \epsilon_{t-1})dt + \sigma_\epsilon d\xi_t \tag{19}$$

We can estimate this equation using linear regression. For a mean reverting series, the expected sign of θ is negative. Then the half-life of the spread process if calculated as, $HL = -\frac{log(2)}{\theta}$. After selecting a set of pairs using our Hurst criteria, we calculate the half-life of the spreads of these pairs for a final filtering process. For our study, We keep all pairs with an in-sample half life less than the trading period, which is 126 days.

This section concludes the pairs selection process. Next, we outline the spread calculation methodology for the final list of pairs selected.

4 Pair Spreads Calculation and Primary Model Methodology

In this section, we present our methodology to compute pairs spreads to generate our trading signal. The traditional approach of computing spreads is done by computing a hedge ratio for the pairs in question. Given, two stocks, i and j, for each unit of i, the hedge ratio will inform us on how many units of j should we short. This can be done by simply running a regression on the log prices as denoted in (7). The coefficient β is the hedge ratio. Then, the spread, denoted as s_t can is simply the residuals obtained from the regression,

$$s_t = log(P_{i,t}) - \beta log(P_{j,t}) \tag{20}$$

However, there are a few drawbacks to this approach. The hedge ratios computed based on linear regression is not symmetric. In the case where are interested in computing a dollar neutral portfolio, with $\beta > 1$ (or $\beta < 1$), we need to long $1/\beta$ (or 1) units of i for every 1 (or $1/\beta$) unit of j. This implies that the hedge ratio β is not symmetric. To understand this, say we swap the legs and run the following regression,

$$log(P_{j,t}) = \gamma + \phi log(P_{i,t}) + \epsilon_t \tag{21}$$

For a hedge ratio to be consistent, one might expect $\phi = 1/\beta$. However, that is not really the case. The reason behind the asymmetry is due to the nature of the loss function for linear regression. The loss function for linear regression is trying to minimize the squared distance between the dependent and the fitted line generated as a function of the predictors. Thus, a linear regression ignores the variations of the predictors since it's only concerned with explaining the variation of the dependent feature. In order to obtain a symmetric fit, one needs to minimize the errors orthogonal to our fitted line. So for hedge ratio calculation, we use an alternate to linear regression known as total least squares [Teetor (2011)]. In the next subsection, we briefly explain total least squares methodology and how to come up with consistent hedge ratios.

4.1 Computing Spreads using Total Least Squares

Let $\boldsymbol{X}$ be $T \times K$ matrix of K assets of log prices comprising the short leg of the spread and $\boldsymbol{y}$ be a $T \times 1$ vector of log prices of the asset on the long end of the spread. In the case of pairs trade, $K = 1$. In the general ordinary least squares framework, where we are concerned about minimizing the error of unexplained

variance of $\boldsymbol{y}$, we can write the following relationship,

$$\boldsymbol{y} + \boldsymbol{s_y} = \boldsymbol{X\Phi} \tag{22}$$

Here, $\boldsymbol{\Phi}$ is a $K \times 1$ vector of coefficients and $\boldsymbol{\epsilon_y}$ is a $T \times 1$ vector of residuals. Now, in the case where both the dependent and independent features are measured with error, we can express the linear relationship as,

$$\boldsymbol{y} + \boldsymbol{s_y} = [\boldsymbol{X} + \boldsymbol{S_X}]\, \boldsymbol{\Phi} \tag{23}$$

$\boldsymbol{S_X}$ is a $T \times K$ matrix residual matrix. The goal of total least squares relates to this aforementioned relationship, to minimize the total measurement error of the entire system, as in for both the independent and dependent variable. Let, $\boldsymbol{Z} = [\boldsymbol{S_X}\ \boldsymbol{s_y}]$. The loss function can be stated as follows,

$$\arg\min ||\boldsymbol{Z}||_F^2;\ s.t.\, \boldsymbol{y} + \boldsymbol{s_y} = [\boldsymbol{X} + \boldsymbol{S_X}]\, \boldsymbol{\Phi} \tag{24}$$

$||\boldsymbol{Z}||_F^2$ is the squared Frobenius norm which can simplified to $\sum_i^{K+1} \sigma_i^2$. σ_i is the i^{th} singular value of $\boldsymbol{Z}$. We can apply Singular Value Decomposition to the system in order to estimate total least squares. The singular value decomposition of the data matrix $[\boldsymbol{X}\ \boldsymbol{y}]$ as,

$$[\boldsymbol{X}\ \boldsymbol{y}] = \boldsymbol{U\Sigma V'} \tag{25}$$

$\boldsymbol{V}$ can be written as,

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V_{XX}} & \boldsymbol{v_{Xy}} \\ \boldsymbol{V_{yX}} & v_{yy} \end{bmatrix} \tag{26}$$

The total least squares solution, $\boldsymbol{\Phi}$ is estimated as,

$$\boldsymbol{\Phi} = -\boldsymbol{v_{Xy}} v_{yy}^{-1} \tag{27}$$

Next, with estimated set of coefficients, We compute the vector of spreads as follows,

$$\boldsymbol{s_y} = \boldsymbol{y} - \boldsymbol{X\Phi} \tag{28}$$

For our framework, we estimate conditional hedge ratios using an expanding window (starting from training period) and then compute pairs spreads using the estimated coefficients. Using an expanding window instead of a static measure ensures that the most recent information is incorporated and position sizing capture the time varying risk associated with the pairs in question.

4.2 Primary Model Trading Signal

Following [Avellanda and Leo, (2010)], for a given pair, we define the score computed from spreads as,

$$\zeta_t = \frac{s_t - EWMA(s_{t-1}, span)}{EWSD(s_{t-1}, span)} \tag{29}$$

$EWMA(s_{t-1}, span)$ and $EWSD(s_{t-1}, span)$ is the exponential weighted moving average and standard deviation of the spread with a given parameter span which is the lookback period. We use a span of 3 months (63 days) to capture more point-in-time variations. The score measures the distance of pair spreads from its

equilibrium, in units of risk. The score can then be used to set our trading signal,

$$\begin{gathered}\text{- Long the spread if } \zeta_t < -1.50;\ (X_t = +1)\\ \text{- Short the spread if } \zeta_t > 1.50;\ (X_t = -1)\\ \text{- Close long if } \zeta_t > -0.50;\ (X_t = 0)\\ \text{- Close short if } \zeta_t < 0.75;\ (X_t = 0)\end{gathered}$$

Here, X_t is the trading status at time t. To explain the rules in detail, for any given pair, we start off with no position and monitor ζ_t. We take a long position on the spread ($X_t = +1$) the moment ζ_t is less than 1.5σ below. We keep on holding on our long position ($X_t = +1$) until ζ_t crosses -0.50σ ($X_t = 0$). A short position ($X_t = -1$) is taken when ζ_t crosses 1.5σ and is held until ζ_t is less than 0.75σ ($X_t = 0$).

4.3 Calculating Strategy Returns

For any pair of stocks i and j at time t, the spread portfolio with $\phi_t < 1$, can be constructed as,

$$\Pi_t = log(P_{i,t}) - \phi_t log(P_{j,t}) \tag{30}$$

ϕ_t is the hedge ratio at time t. The dollar neutral spread portfolio return from period t to $t+1$ can be written as,

$$\begin{aligned}\Pi_{t+1} - \Pi_t &= [log(P_{i,t+1}) - \phi_{t+1} log(P_{j,t+1})] - [log(P_{i,t}) - \phi_t log(P_{j,t})]\\ &= \underbrace{[log(P_{i,t+1}) - log(P_{i,t})]}_{\text{Long Leg Return}} + \underbrace{[\phi_t log(P_{j,t}) - \phi_{t+1} log(P_{j,t+1})]}_{\text{Short Leg Return}}\\ &= R_{s,t+1}\end{aligned} \tag{31}$$

In cases where $\phi_t > 1$, the general equation will stay the same, with the exception that now for each unit of short position in j, we will long position of $1/\phi_t$ in i. The return for the relative value strategy, $R_{rv,t+1}$, can be written as,

$$R_{rv,t+1} = X_t * R_{s,t+1} \tag{32}$$

5 Generating Meta Labels for Primary Model

5.1 Meta Labelling Methodology

A common caveat of rules-based signal generation models in finance (like the one described in section 4.2) is although it provides us with prediction for our strategy, it doesn't provide us any measure of confidence/likelihood of whether this predicted signal will lead to profitable outcomes. Thus, having a probabilistic approach to inform us on the confidence in our signals can be crucial to improve risk adjusted returns of the strategy. We refer to this secondary model as Meta Labelling [López de Prado (2018)].

To understand how meta labelling can improve upon the predictability of our primary signal generation model, we refer to López de Prado (2018). Consider a primary model which produces T IID bets each year, with X_t being the trading signal/status derived from our rule based approach. Now in the case of binary outcomes, let's define the positive payoff r_+ with probability p and the negative payoff r_- with probability $1 - p$, with $r_+ > r_-$. The expected profit of the strategy from one signal can be written as,

$$\begin{aligned}\mathbb{E}\left[X_t\right] &= pr_+ + (1-p)r_- \\ &= (r_+ - r_-)p - r_-\end{aligned} \tag{33}$$

The variance of the strategy from one signal can be written as,

$$\begin{aligned}\mathbb{V}[X_t] &= \mathbb{E}[X_t^2] - \mathbb{E}[X_t]^2 \\ &= (r_+ - r_-)^2 p(1-p)\end{aligned} \tag{34}$$

The annualized Sharpe ratio of the strategy can be written as,

$$S.R. = \frac{(r_+ - r_-)p - r_-}{(r_+ - r_-)\sqrt{p(1-p)}}\sqrt{T} \tag{35}$$

If the strategy is generally unprofitable, i.e. the positive payoff is less than the absolute values of negative payoffs, $r_+ << -r_-$, it may still be possible to increase the Sharpe ratio of the strategy by improving the probability with the expense of T. In order to understand improving p leads to better risk adjusted returns, let us understand the key components of a confusion matrix. Table 1 shows an illustration of the confusion matrix.

Table 1: Confusion Matrix

		$\hat{y}$	
		1	0
y	1	True Positive (TP)	False Negative (FN)
	0	False Positive (FP)	True Negative (TN)

$\hat{y}$ is the predicted outcome from our primary model and y is the realized outcome of the asset. The elements inside the matrix are self-explanatory. Next, we define two important metrics, Precision and Recall.

$$Precision = \frac{TP}{TP + FP} \tag{36}$$

$$Recall = \frac{TP}{TP + FN} \tag{37}$$

Precision tells us out of all the predicted outcomes, what percentage of the predictions were profitable. On the other hand, Recall denotes what percentage of profitable outcomes are predicted accurately by our model. An investment strategy where the magnitude of returns are larger on the false positive outcomes as opposed to the true positive outcomes (typically the case in most trading strategies) can lead to an overall unprofitable strategy. This is why it is important to quantify the likelihood of our primary model predictions to ensure we are not getting caught up in false positive outcomes with large market moves. The overall profitability can be improved by giving up some of the recall of the primary model to improve its precision, thus lowering false positives and improving true negatives. Supervised classification algorithms such as, Random Forest, Gradient Boosting and Neural Network etc. can aid us in estimating probabilities for our primary models and use the meta labels derived from the probabilities as an risk management overlay to our original signals.

Our meta labelling model relies of estimating probabilities from an ensemble of popular machine learning classifiers. The next subsection we outline the classifiers chosen, the feature set used for estimation, and the hyper parameter space for each classifier over which we will conduct cross validation procedure.

5.2 Overview of Meta Labelling Classifiers

In this section, we provide an overview of the classifiers used to develop our meta labelling model. We primarily focus on three different classes of models popularized in supervised machine learning space, notably bagging, boosting and neural network models. We choose Random Forest and a 3-Layer Neural Network for bagging classifier and neural net classifier respectively. For the boosting classifiers, we choose AdaBoost and two popular gradient boosting algorithms, namely XGBoost and LightGBM, available in open source environment. We choose to include both XGBoost and LightGBM given their distinctive differences in algorithm with regards to how each grows the weak learner (Decision Tree for example). This can lead to different outcomes despite having similar differentiable loss functions for optimization. The general framework for our classifiers can be written as,

$$p_{c,t+1} = Prob_c(y_{t+1} = 1|\boldsymbol{f_t}) = G^c(\boldsymbol{f_t}) \tag{38}$$

$$y_{t+1} = \begin{cases} 1; & if\ sign(R_{rv,t+1}) = X_t \\ 0; & otherwise \end{cases} \tag{39}$$

$G^c(.)$ is a generalized functional form of the cth classifier and $\boldsymbol{f_t}$ is a $P \times 1$ vector of features. The primary features for the model are the scores ζ_t and derived trading signal X_t. We also generate an additional set of features from the raw spreads which are unrelated to our primary model and stack them alongside the primary features. The additional set of features are listed below,

Table 2: Additional Features for Meta Label Classifiers

	Volatility	Skewness	Kurtosis	Hurst Exponent	AutoCorrelation
Lookbacks	[21, 63]	[21, 63]	[21, 63]	[21, 63]	[21]
Lags					[1, 5, 10]

For each pair, we estimate all five classifiers in the training periods. We define the ensemble forecast, $\hat{y}_{t+1}$, as,

$$\hat{y}_{t+1} = \begin{cases} 1; & if\ \left[\frac{1}{C}\sum_c^C \mathbb{1}_{(p_{c,t+1}>0.50)}\right] > 0.50 \\ 0; & otherwise \end{cases} \tag{40}$$

To apply the meta labelling prediction during our trading period, we can modify (32) and rewrite our out of sample (oos) strategy returns as,

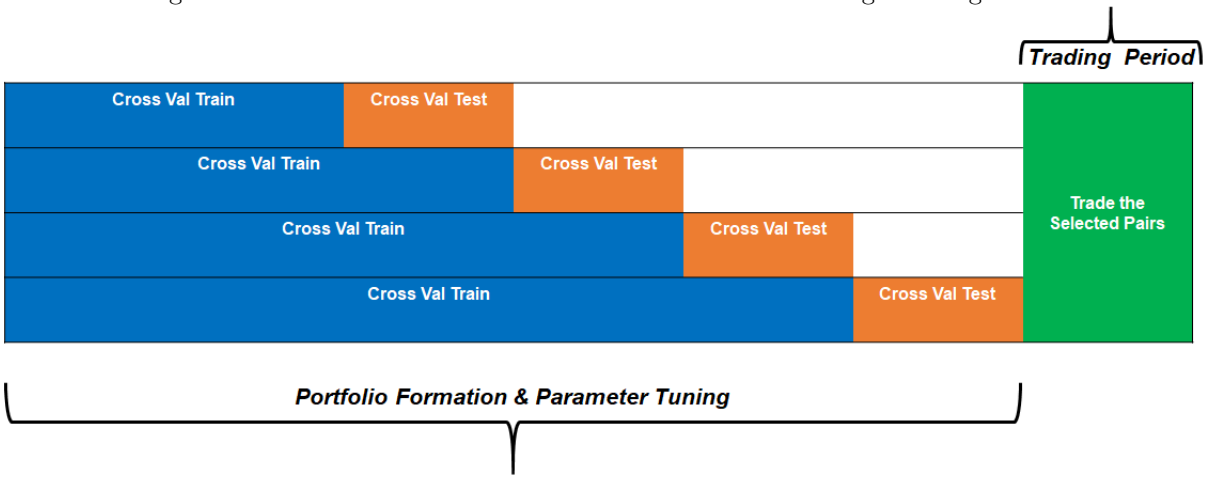
$$R^{oos}_{rv,t+1} = \hat{y}^{oos}_{t+1} * X^{oos}_t * R^{oos}_{s,t+1} \tag{41}$$

The general idea is that for any given pair, if the predicted probabilities from our ensemble classfiers predict higher likelihood of profitability i.e. $\hat{y}_{t+1} = 1$, we continue trading based on our primary model signal. We close our position for any trading day if the meta labels suggest lower likelihood of profitability, i.e. $\hat{y}_{t+1} = 0$.

5.3 Hyperparameter Tuning and Cross Validation Procedure

In this section we briefly discuss the cross validation procedure for our machine learning classifiers. Given the nature of time series data, we avoid the standard k-fold approach to cross validation where the reshuffling of our training set will potentially break the temporal dependency of time series data. In order to preserve the dependency, we conduct the k-fold process on a rolling basis. The general idea is start off with small set of data from the beginning of the training period and fit our classifiers, then test its performance on later data points. These validation data sets are then added to the training set for the next iteration of the cross validation procedure on a rolling basis. We select the number of cross validation splits to be 4. The figure below provides an illustration of such procedure,

Figure 2: An Illustration of Cross Validation Procedure during Training Period



We use a random search process for hyperparameter tuning. For a given classifier, the tuning process will randomly select a combination of hyperparameters from the defined hyperparameter distribution and conduct k-fold cross validation. The procedure calculates a performance metric for each possible combination and picks the optimal combination as the one with the highest average score across all cross validation splits. We use Negative Log Loss (NLL) as our performance metric,

$$NLL = -\frac{1}{t_v}\sum_{i}^{t_v}[y_i log(\hat{y_i}) + (1-y_i)log(1-\hat{y_i})] \tag{42}$$

t_v is the number of samples in the validation set, y_i is realized indicator and $\hat{y_i}$ is the predicted probability for the given validation observation. For each classifier, we select a list hyperparameters to estimate. We pick these lists based on objective judgement, our knowledge on each classification algorithm and how much each parameter will be more sensitive to the overall prediction.

The table below lists out the hyperparameters chosen to optimize for each classifier,

Table 3: Hyperparameters for All Classifiers

HyperParameters	Random Forest	AdaBoost	XGBoost	LightGBM	3L-Neural Network
n_estimators	[250, 500, 750, 1000]	[250, 500, 750, 1000]	[250, 500, 750, 1000]	[250, 500, 750, 1000]	
max_features	uniform(0.1, 1)				
min_samples_leaf	[1, 5, 10, 15, 30]				
ccp_alpha	uniform(0.001, 0.5)				
learning_rate		uniform(0.01,0.5)	uniform(0.01,0.5)	uniform(0.01,0.5)	['constant', 'invscaling', 'adaptive']
max_depth			[3, 5, 9, 11, 13, 15]	[3, 5, 9, 11, 13, 15]	
subsample			[0.1, 0.25, 0.5, 0.75, 1.0]	[0.1, 0.25, 0.5, 0.75, 1.0]	
min_child_weight			uniform(0.01,1)	uniform(0.01,0.5)	
num_leaves				[16, 32, 64, 128]	
hidden_layer_sizes					[(25, 25, 25), (50, 50, 50), (75, 75, 75), (100, 100, 100)]
alpha					uniform(0.01,0.5)